StructMem: Structured Memory for Long-Horizon Behavior in LLMs

Anonymous ACL submission

Abstract

Long-term conversational agents need memory systems that capture relationships between events, not merely isolated facts, to support temporal reasoning and multi-hop question answering. Current approaches face a fundamental trade-off: flat memory is efficient but fails to model relational structure, while graph-based memory enables structured reasoning at the cost of expensive and fragile construction. To address these issues, we propose **StructMem**, a structure-enriched hierarchical memory framework that preserves event-level bindings and induces cross-event connections. By temporally anchoring dual perspectives and performing periodic semantic consolidation, StructMem improves temporal reasoning and multi-hop performance on LoCoMo, while substantially reducing token usage, API calls, and runtime compared to prior memory systems.

1 Introduction

Persistent memory systems are essential for language model agents to maintain coherence in long-term interactions (Park et al., 2023). Beyond factual recall, long-horizon dialogue requires reasoning over temporal dependencies, causal chains, and multi-hop relationships across turns (Weller et al., 2025; Huang et al., 2025; Maharana et al., 2024; Wu et al., 2024; Yang et al., 2018). This necessitates memory representations that organize events into temporally grounded and relational structures (Kwiatkowski et al., 2019).

Existing memory systems largely fall into two paradigms, flat memory and graph memory, exhibit a trade-off between efficiency and structured reasoning, as illustrated from Figure 1. Specifically, flat memory systems (Fang et al., 2025a; Zhong et al., 2024; Packer et al., 2023) store facts or summaries as independent units, but fail to preserve cross-event relations, causing retrieval over long histories to degrade into shallow similarity matching (Liu et al., 2023; Zhuang et al., 2025). Graph-based systems (Chhikara et al., 2025; Rasmussen et al., 2025) recover relational structure via entity–relation extraction, but incur high construction cost, require cascaded inference (Edge et al., 2024), and are vulnerable to error accumulation from noisy extractions (Zhuang et al., 2025). We argue that these limitations arise from an inappropriate memory unit. Rather than isolated facts or triplets, the fundamental unit of conversational memory should be a *temporally grounded relational event*, which preserves causal and interpersonal context without rigid schemas.

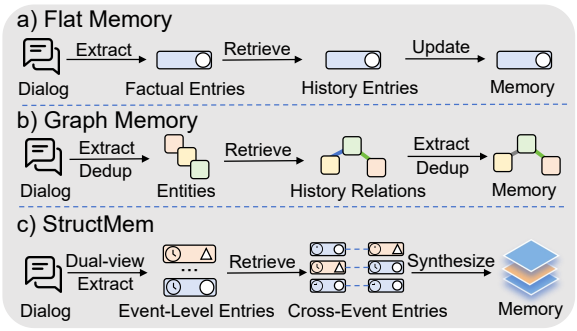


Figure 1: Three paradigms of Memory systems.

Based on this insight, we propose **StructMem**, a hierarchical memory framework built around event-centric representations. This abstraction preserves both what happened and how events relate across agents and time, while avoiding explicit schema design, entity resolution, and symbolic graph traversal. Specifically, at the event level, StructMem constructs structured episodes through dual-perspective extraction, capturing both event content and interactional relations within temporal context. At the cross-event level, it performs periodic consolidation over semantically related events, exploiting temporal locality to efficiently induce higher-level relational structure. Experiments on LoCoMo show that StructMem improves long-horizon reasoning while significantly reducing computational overhead.

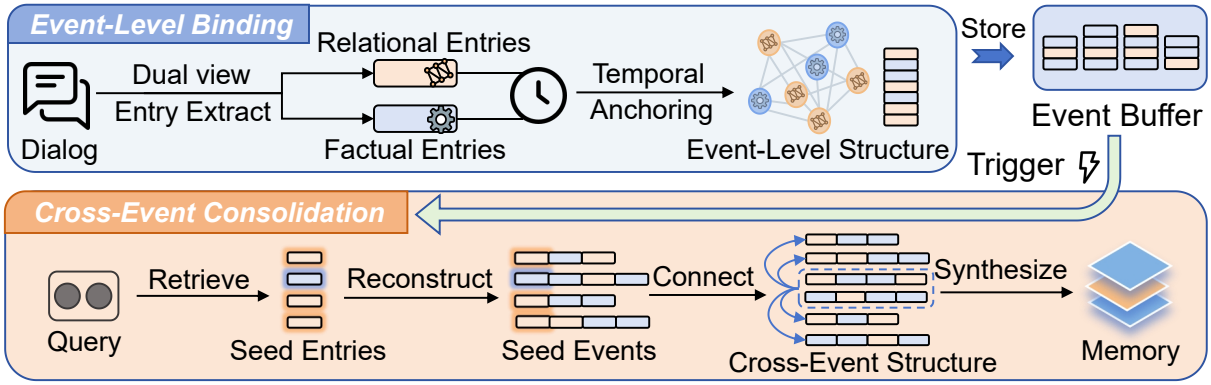


Figure 2: StructMem's hierarchical memory organization. **Event-Level Binding** constructs event-level structure by extracting dual perspectives and anchoring them temporally. **Cross-Event Consolidation** constructs cross-event structure through semantic retrieval, event reconstruction, and consolidation synthesis.

2 Method

We propose **StructMem**, a framework that achieves structure-enriched organization through hierarchical design. The framework operates at two levels: event-level structure (§2.1) preserves relational bindings within utterances, while cross-event structure (§2.2) connects information across temporal boundaries.

2.1 Event-Level Binding

Event-level binding preserves the connection between factual content and relational context within individual utterances through dual-perspective extraction and temporal anchoring.

Dual-Perspective Extraction. For each utterance m_i in the dialogue stream, we extract entries from two complementary perspectives using language model $\mathcal{L}$ with prompts P_{fact} and P_{rel}:

$$\Phi_i \cup \Psi_i = \mathcal{L}(P_{fact} \| m_i) \cup \mathcal{L}(P_{rel} \| m_i), \quad (1)$$

where $\Phi_i = \{c_{i,1}, \ldots, c_{i,j}\}$ contains *factual entries* describing event content, and $\Psi_i = \{r_{i,1}, \ldots, r_{i,k}\}$ contains *relational entries* capturing interpersonal dynamics, causal influences, and temporal dependencies.

By representing both in natural language rather than rigid triplets, we preserve the contextual nuances required for episodic grounding while avoiding entity resolution overhead.

Temporal Anchoring. To preserve the binding between relational and factual information, all entries are anchored to their originating timestamp τ_i, forming an event-level unit:

$$\mathcal{M} \leftarrow \bigcup_{i=1}^{N} \{\langle x, \mathbf{e}_x, \tau_i \rangle \mid x \in \Phi_i \cup \Psi_i\}, \quad (2)$$

where $\mathbf{e}_x$ denotes the embedding of entry x. This temporal coupling enables reconstruction of complete factual-relational events during retrieval.

2.2 Cross-Event Consolidation

Cross-event consolidation connects information across temporal by periodically synthesizing semantically related events. We trigger synthesis when accumulated events exceed a time threshold.

Semantic Event Connections. We buffer unconsolidated entries since the last consolidation. The buffered entries are temporally ordered:

$$\mathcal{C}_{buf} = \text{Sort}_\tau\{x \in \mathcal{M}_{buffer}\}, \quad (3)$$

where $\mathcal{M}_{buffer}$ denotes the buffered entries. We encode the buffered context into an aggregated query by concatenating all buffered entry texts and encoding them with embedding model. We then rank all historical entries by cosine similarity to this query and retrieve the top-K most semantically similar entries as seeds, denoted as $\mathcal{S}_k$..

For each seed entry $x^* \in \mathcal{S}_k$, we reconstruct its complete event context by retrieving all entries sharing the same timestamp:

$$E_\tau(x^*) = \{x' \in \mathcal{M} \mid \tau(x') = \tau(x^*)\}. \quad (4)$$

These reconstructed events, along with buffered events, form the cross-event structure based on semantic relevance.

$$\mathcal{C}_{cross} = \mathcal{C}_{buf} \cup \bigcup_{x^* \in \mathcal{S}_k} E_\tau(x^*). \quad (5)$$

Memory Consolidation through Synthesis. Unlike conventional summarization that performs

Method	Overall ↑	Performance by Type ↑				Build Tokens (M) ↓			Calls ↓	Time (s) ↓
		Multi	Open	Single	Temp	In	Out	Sum		
OpenAI	71.82	69.86	53.12	84.66	45.48	–	–	–	–	–
FullText	73.83	68.79	56.25	**86.56**	50.16	–	–	–	–	–
MiniRAG	63.51	56.74	58.33	75.74	38.94	9.022	1.081	10.103	2508	**2566**
LightRAG	68.83	66.31	50.00	77.53	53.89	10.014	1.916	11.931	13576	60469
LangMem	58.10	62.23	47.92	71.12	23.43	9.873	1.192	11.066	5990	26281
A-Mem	64.16	56.03	31.25	72.06	60.44	9.126	2.368	11.494	11754	60607
Mem0	66.88	67.13	51.15	72.93	59.19	10.958	1.239	12.196	9181	30057
MemoryOS	58.25	56.74	45.83	67.06	40.19	1.889	0.939	2.868	5534	24220
Mem0^g	68.44	65.71	47.19	75.71	58.13	33.512	2.313	35.825	53514	115670
Zep	75.14	**74.11**	**66.04**	79.79	67.71	–	–	–	–	–
Memobase	75.78	70.92	46.88	77.17	**85.05**	–	–	–	–	–
StructMem	**76.82**	68.77	46.88	81.09	81.62	**1.501**	**0.436**	**1.937**	**1056**	22854

Table 1: Performance and resource consumption comparison of memory systems on LoCoMo dataset. ↑: larger is better; ↓: smaller is better. **The best results** are marked in bold, the second-best results are underlined. Row colors distinguish method categories: RAG methods , Flat Memory methods , and Structural Memory methods . OpenAI and FullText have no construction cost; Zep and Memobase do not expose construction details.

lossy compression on sequential text, our consolidation mechanism operates on semantically-reconstructed event clusters. It explicitly synthesizes cross-event relational hypotheses, forming a complementary abstraction layer that enables multi-hop reasoning while preserving the fidelity of raw episodic memory.

$$\mathcal{M} \leftarrow \mathcal{C}_{cons} = \mathcal{L}(P_{cons} \| \mathcal{C}_{cross}). \quad (6)$$

3 Experiments

3.1 Experimental Setup

Dataset and Metrics. We evaluate on the LoCoMo benchmark (Maharana et al., 2024) (see details in Appendix A.3). Effectiveness is measured using LLM-as-a-judge evaluation; efficiency is measured by token usage, API calls, and runtime during memory construction.

Baselines. We compare against RAG-based systems (OpenAI, FullText, MiniRAG, LightRAG) and memory-based systems categorized into flat memory (LangMem, A-Mem, Mem0) and structural memory (MemoryOS, Mem0^g, Zep, Memobase). All methods use gpt-4o-mini as the backbone and text-embedding-3-small for embeddings.

3.2 Overall Performance

Table 1 shows StructMem achieves state-of-the-art overall performance on LoCoMo, with substantial gains in multi-domain and temporal reasoning where cross-event connections are critical for understanding causal relationships across dialogue sessions. Beyond effectiveness, StructMem demonstrates exceptional efficiency: compared to existing memory systems, it reduces token consumption and requires significantly fewer API calls, as our progressive structural organization avoids the expensive post-hoc graph construction.

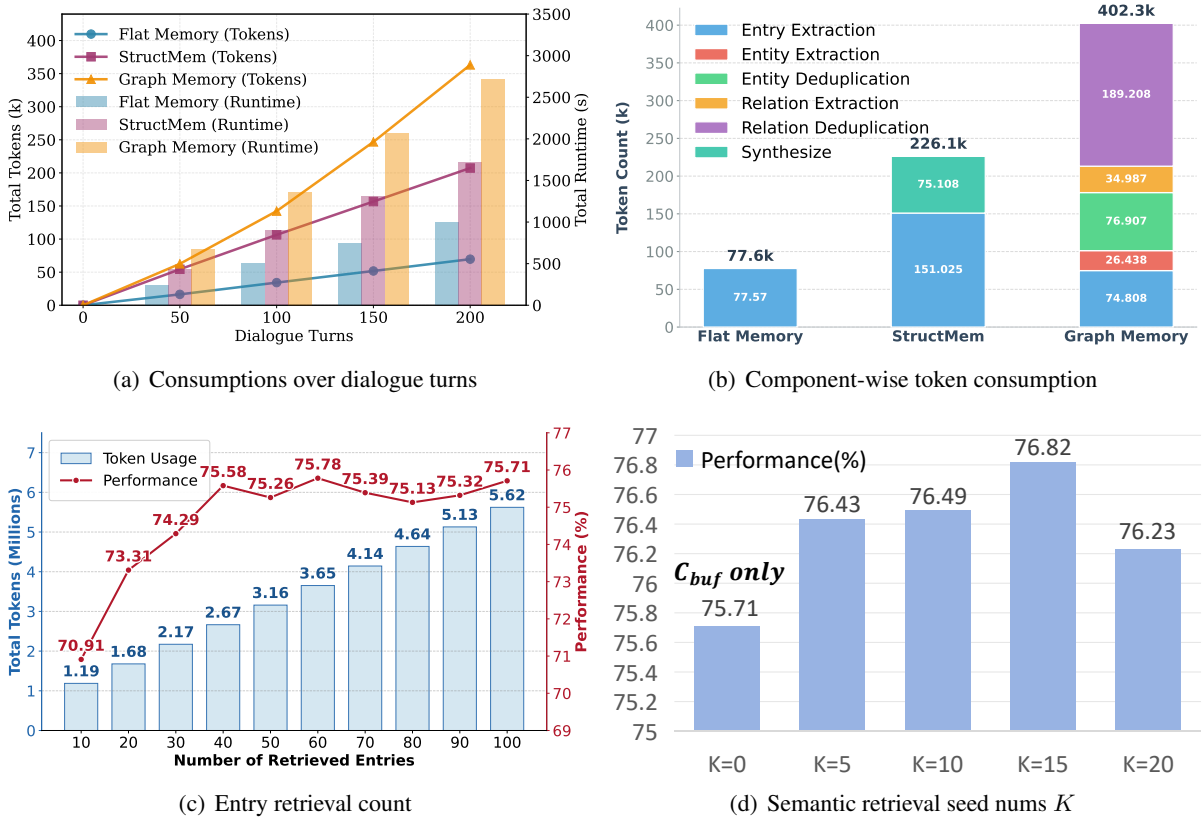
3.3 Analysis

Paradigm Comparison. We implement all three paradigms on the LightMem framework to systematically compare these paradigms through both effectiveness and efficiency.

Method	Multi	Open	Single	Temp
Flat Memory	66.31	46.88	78.83	78.50
Graph Memory	66.67	48.96	80.50	76.64
w/o Cross-Event	66.31	46.88	80.86	79.44
StructMem	68.77	46.88	81.09	81.62

Table 2: Paradigm comparison and ablation study on LoCoMo dataset.

To validate the effectiveness of each paradigm, we conduct studies in Table 2. Starting from Flat Memory as the baseline, Graph Memory achieves improvements on single-session and open-domain tasks, though it decreases on temporal reasoning. In contrast, our approach demonstrates consistent improvements across all task types. Event-level structure improves performance in temporal rea-



(a) Consumptions over dialogue turns

(b) Component-wise token consumption

(c) Entry retrieval count

(d) Semantic retrieval seed nums K

Figure 3: Analysis of efficiency across memory paradigms and internal mechanisms of StructMem.

soning and single-session. Cross-event structure yields further gains by capturing cross-temporal causal relationships.

To examine computational efficiency, we analyze token usage and runtime on the first conversation of LoCoMo. Figure 3(a) shows that Graph Memory incurs significantly higher token usage and runtime as dialogue progresses. Figure 3(b) reveals the source: graph construction requires four cascading LLM operations per event, with deduplication overhead growing quadratically. In contrast, StructMem achieves efficiency through buffered consolidation: by exploiting temporal locality, in which semantically related events naturally cluster within short time windows, the system accumulates events and processes them in batch during periodic synthesis. This reduces cross-event organization from per-event operations to periodic batch processing, substantially cutting API calls and token consumption.

StructMem Internal Mechanisms. We analyze whether hierarchical organization provides genuine reasoning gains beyond retrieval scaling. Figure 3(c) reveals that flat retrieval performance peaks at 60 entries and plateaus thereafter, indicating that simply retrieving more atomic entries cannot improve effectiveness, as the bottleneck is knowledge reasoning rather than coverage. Figure 3(d) demonstrates how cross-event structure transcends this ceiling by creating new information: without event connections ($K = 0$), performance matches flat retrieval's plateau, but introducing cross-event mechanism yields substantial gains. This breakthrough, unattainable through flat scaling, shows that hierarchical consolidation reconstructs causal relationships across temporal, enabling fundamentally new reasoning capabilities.

4 Conclusion

We propose StructMem, which achieves structure-enriched organization through hierarchical design: preserving event-level bindings and enabling cross-event consolidation, StructMem preserves temporal and relational structures without the computational overhead of continuous graph maintenance. Experiments on LoCoMo demonstrate that StructMem achieves better performance with strong results in multi-hop and temporal reasoning, while substantially reducing token consumption, API calls, and runtime compared to prior memory systems.

Limitations

Despite its strong performance, StructMem has several limitations. The quality of dual-perspective extraction is highly dependent on instruction prompts, where suboptimal design may result in incomplete or inaccurate relational information capture. Future research could investigate automated prompt optimization to improve robustness across various dialogue contexts. Additionally, the framework primarily addresses memory expansion and synthesis but currently lacks an explicit mechanism for conflict resolution and memory updating. As user facts or preferences may evolve over long horizons, the absence of a revision process could lead to inconsistencies between historical summaries and new information. Future iterations should incorporate memory decay or updating strategies to ensure the hierarchical organization accurately reflects the most current state of the interaction.

References

Vinay Chaudhri, Chaitanya Baru, Naren Chittar, Xin Dong, Michael Genesereth, James Hendler, Aditya Kalyanpur, Douglas Lenat, Juan Sequeda, Denny Vrandečić, and 1 others. 2022. Knowledge graphs: introduction, history and, perspectives. *AI Magazine*, 43(1):17–29.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Cody V Dong, Qihong Lu, Kenneth A Norman, and Sebastian Michelmann. 2025. Towards large language models with human-like episodic memory. *Trends in Cognitive Sciences*.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. 2025. Memr^3: Memory retrieval via reflective reasoning for llm agents. *Preprint*, arXiv:2512.20237.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, and 1 others. 2025a. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2025b. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Zhengjun Huang, Zhoujin Tian, Qintian Guo, Fangyuan Zhang, Yingli Zhou, Di Jiang, and Xiaofang Zhou. 2025. Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning. *arXiv preprint arXiv:2511.01448*.

Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Soumen Chakrabarti, and 1 others. 2020. Openie6: Iterative grid labeling and coordination analysis for open information extraction. *arXiv preprint arXiv:2010.03147*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Mo Li, L. H. Xu, Qitai Tan, Long Ma, Ting Cao, and Yunxin Liu. 2025. Sculptor: Empowering llms with cognitive agency via active context management. *Preprint*, arXiv:2508.04664.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. On the theoretical limitations of embedding-based retrieval. *Preprint*, arXiv:2508.21038.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*.

Siyu Xia, Zekun Xu, Jiajun Chai, Wentian Fan, Yan Song, Xiaohan Wang, Guojun Yin, Wei Lin, Haifeng Zhang, and Jun Wang. 2025. From experience to strategy: Empowering llm agents with trainable graph memory. *arXiv preprint arXiv:2511.07800*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 50–61.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. 2025. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. *arXiv preprint arXiv:2510.10114*.

A Appendix

A.1 License

This work uses the LoCoMo benchmark dataset, which is publicly available for academic research purposes. We follow all usage terms specified by the dataset authors.

A.2 Background

Long-term memory serves as the cognitive foundation for agents to maintain persona consistency and perform reasoning across extended horizons (Maharana et al., 2024; Wu et al., 2024; Dong et al., 2025; Huang et al., 2025).

Early approaches addressed the context window limitation by externalizing history into flat vector databases (Park et al., 2023; Packer et al., 2023; Zhong et al., 2024). While efficient for semantic matching, this paradigm fundamentally treats interaction history as an unordered bag of propositions, severing the temporal progression, causal dependencies, and relational substrate that bind events into coherent narratives (Gao et al., 2023; Liu et al., 2023). This flat representation leads to fragmented retrieval where isolated facts are returned without the contextual scaffolding necessary for complex reasoning (Weller et al., 2025; Li et al., 2025). Recent work has explored enhanced retrieval strategies through reflective reasoning and closed-loop control mechanisms (Du et al., 2025), yet these improvements still operate within the fundamental constraints of flat representations. Even with extended context windows, flat systems suffer from the Lost-in-the-Middle phenomenon (Liu et al., 2023), where attention mechanisms degrade in ultra-long sequences, reducing multi-hop reasoning to superficial similarity search (Zhuang et al., 2025).

To bridge this reasoning gap, the field has increasingly pivoted towards structure-enriched architectures, particularly those leveraging Knowledge Graphs. Static graph approaches, such as Microsoft GraphRAG (Edge et al., 2024) and HippoRAG (Gutiérrez et al., 2025), employ hierarchical community detection and Personalized PageRank to facilitate global sense-making and multi-hop traversal. Concurrently, dynamic memory systems tailored for agents, such as Mem0^g (Chhikara et al., 2025) and Zep (Rasmussen et al., 2025), have introduced evolving schemas to capture the fluidity of user interactions. Recent advances further explore trainable graph representations that enable agents to learn strategic meta-cognition from past experiences (Xia et al., 2025) and lightweight hierarchical graphs with entity-relation indexing for efficient long-term reasoning (Huang et al., 2025). These graph-based architectures have demonstrated substantial improvements in multi-agent collaboration (Zhang et al., 2025) and procedural skill reuse (Fang et al., 2025b), validating the necessity of structural organization.

Despite these advancements, imposing explicit graph structures on natural dialogue introduces inherent trade-offs. Compressing fluid, nuanced narratives into rigid entity-relation triplets often incurs semantic loss, stripping away the subtle contextual

Reasoning Type	# Questions
Single-hop	841
Multi-hop	282
Temporal	321
Open-domain	96

Table 3: Statistics of LoCoMo questions used.

qualifiers and discourse continuity necessary for episodic grounding (Chaudhri et al., 2022; Zhuang et al., 2025). This compression challenge is compounded by extraction instability: hallucinated relations can propagate as persistent structural noise, calcifying errors into the memory topology (Zhong and Chen, 2021; Kolluru et al., 2020). Beyond these quality concerns, the computational overhead required for continuous graph maintenance, including entity resolution, relation deduplication, and conflict detection, poses latency challenges for real-time agentic applications (Edge et al., 2024; Fang et al., 2025a).

A.3 Dataset

We evaluate on the **LoCoMo** benchmark (Maharana et al., 2024), which contains 10 long-term conversations with an average of 588 turns and 16,618 tokens per conversation. We focus on the question answering task, utilizing four reasoning types from the benchmark. Table 3 shows the statistics of questions used in our evaluation. Model performance is evaluated using LLM-as-a-judge.

A.4 Implementation Details

We provide key implementation details for StructMem to facilitate reproducibility. **Memory construction:** We set the time window threshold to 1 hour for triggering consolidation. For cross-event consolidation, we retrieve top-15 semantically similar seed entries from historical memory. **Question answering:** During inference, we retrieve 60 entries and 5 synthesis from memory to provide context for answer generation.

A.5 Prompt Templates

We present the prompt templates used for memory construction, question answering, and evaluation in StructMem.

For memory construction, we design prompts for different paradigms implemented in the LightMem framework. For Flat Memory, the factual entry extraction prompt (Figure 4 and Figure 5) guides the model to decompose utterances into objective event descriptions. For StructMem, the relational entry extraction prompt (Figure 6 and Figure 7) instructs the model to capture interaction dynamics, causal influences, and temporal dependencies. The narrative synthesis prompt (Figure 8) consolidates local and retrieved contexts into coherent summaries during Macro Synthesis. For Graph Memory, the entity extraction prompt (Figure 9) identifies key entities from dialogue. The entity deduplication prompt (Figure 10) normalizes extracted entities to eliminate redundancy. The relation extraction prompt (Figure 11) constructs connections between entities. The relation deduplication prompt (Figure 12) resolves contradictions in the knowledge graph.

For question answering, we provide separate prompts tailored to different memory architectures. Figure 13 shows the prompt for StructMem with dual-circuit retrieval that leverages both atomic entries and consolidated summaries. Figure 14 and Figure 15 present prompts adapted for flat memory and graph-based memory baselines, respectively.

For evaluation, we use the LLM-as-a-judge prompt (Figure 16) to assess response correctness and coherence.

A.6 Case Study

Table 4 presents a case study comparing how different memory paradigms handle temporal reasoning over joint participation. The query asks when two speakers attended an event together, requiring inference over co-participation relationships that are not explicitly stated in individual conversational turns.

Flat Memory retrieves factual entries independently: Caroline attended Pride fest "last year" (temporally anchored to August 17, 2023, referring to 2022), while Melanie enjoyed time "with the whole gang" at Pride fest. Without any mechanism to connect these isolated facts, the system concludes "they haven't gone together," failing to recognize the implicit joint participation.

Graph Memory constructs entity-relation triples on top of the same factual entries. While the graph captures individual attendance, these remain isolated nodes without explicit co-participation edges. The post-hoc graph structure cannot infer that mentions of the same event by different speakers within the same conversation indicate joint attendance. Consequently, it produces an incorrect temporal inference: "Last month, June 2023."

StructMem addresses this limitation through

Query	*When did Caroline and Melanie go to a pride festival together?*
Method	**Retrieved Content**
Flat Memory	**Factual Entries:** • Caroline attended pride parade on 2023-08-11 • Caroline had a blast at Pride fest last year (recorded 2023-08-17) • Melanie enjoyed time with the whole gang at Pride fest (recorded 2023-08-17)
Graph Memory	**Factual Entries:** • Caroline attended pride parade on 2023-08-11 • Caroline had a blast at Pride fest last year (recorded 2023-08-17) • Melanie enjoyed time with the whole gang at Pride fest (recorded 2023-08-17) **Entity-Relation Graph:** • caroline → attended → pride_parade • caroline → had_blast_at → pride_fest • melanie → enjoyed_time_at → pride_fest • melanie → expressed_excitement → caroline's_pride_involvement
StructMem	**Event Memory:** • Caroline attended pride parade on 2023-08-11 • Caroline had a blast at Pride fest last year (recorded 2023-08-17) • Melanie showed interest in Caroline's pride parade experience • Melanie enjoyed time with the whole gang at Pride fest (recorded 2023-08-17) • Melanie expressed excitement about Caroline's LGBTQ+ community involvement **Synthesis Memory:** "On August 17, 2023... **As they reminisced about their enjoyable time at Pride fest last year**, Melanie suggested planning a family outing, while Caroline proposed a special outing just for the two of them this summer..."
Prediction	**Flat Memory:** "They haven't gone together." **Graph Memory:** "Last month, June 2023." **StructMem:** "Last year, August 2022." **Reference:** 2022

Table 4: Case study comparing three memory paradigms on joint participation reasoning. Flat Memory and Graph Memory cannot establish co-participation from isolated entries, while StructMem's synthesis correctly identifies shared experiences.

two mechanisms. First, relational entries capture interpersonal dynamics during extraction: "Melanie showed interest in Caroline's pride parade experience" provides crucial context about their shared discussion. Second, synthesis consolidates temporally co-located entries. When Caroline's Pride fest mention appears adjacent to Melanie's in the chronologically sorted context, the relational entry's possessive pronoun "their" signals joint participation. The synthesis then makes this implicit connection explicit: "their enjoyable time at Pride fest last year," enabling the system to correctly answer "Last year, August 2022."

This case demonstrates why extraction-time structural capture outperforms post-hoc graph construction for temporal reasoning. By organizing information hierarchically during memory formation rather than overlaying structure afterward, StructMem preserves the temporal and relational context necessary for inferring implicit relationships across conversational turns.

Prompts for Factual Entry Extract

```
[SYSTEM]: You are a Personal Information Extractor.
Your task is to extract **all possible facts or information** about the speakers from a
conversation,
where the dialogue is organized into topic segments separated by markers like:
--- Topic X ---
[timestamp, weekday] <source_id>.<SpeakerName>: <message>
...
Note: Messages may include visual context marked as [visual_context: ...] which provides
additional scene information.
Important Instructions:
0. You MUST process messages **strictly in ascending source_id order** (lowest → highest).
   For each message, stop and **carefully** evaluate its content before moving to the next.
   Do NOT reorder, batch-skip, or skip ahead — treat messages one-by-one.
1. You MUST process every user message in order, one by one.
   For each message, decide whether it contains any factual information.
   - If yes → extract it and rephrase into a standalone sentence.
   - Do NOT skip just because the information looks minor, trivial, or unimportant.
     Extract ALL meaningful information including:
     * Past events and current states
     * Future plans and intentions
     * Thoughts, opinions, and attitudes
     * Wants, hopes, desires, and preferences
2. **CRITICAL - Preserve All Specific Details**:
   When extracting facts, you MUST include ALL specific entities and details mentioned:
   - **Full names with context**: "The Name of the Wind" by Patrick Rothfuss (not just "a
book")
   - **Complete location names**: Galway, Ireland; The Cliffs of Moher; Barcelona (not just
"a city")
   - **Specific event names**: benefit basketball game, study abroad program (not just "an
event")
   - **Product/item details**: vintage camera, brand new fire truck (not just "a camera")
   - **Numbers and quantities**: 4 years ago, next month, last week
   - **Company/organization names**: beverage company, fire-fighting brigade
   Additionally, **infer implied information** when clearly supported:
   - If multiple related items mentioned → may infer general pattern
   - Keep BOTH specific facts AND inferred insights as separate entries
3. Perform light contextual completion so that each fact is a clear standalone statement.
4. **Time Handling**:
   Note: Distinguish mention time (when said) vs event time (when happened).
   - For events with relative time (yesterday, last week, X ago, next month):
     Preserve the relative time and reference the message timestamp (YYYY-MM-DD).
     Format: "<fact with ALL details> <relative time> <timestamp>."
   - For ongoing/timeless facts: No time annotation needed.
```

Figure 4: Factual entry extraction prompt (Part 1).

Prompts for Factual Entry Extract

```
5. Output format:
   Always return a JSON object with key `"data"`, which is a list of items:
   {
     "source_id": "<source_id>",
     "fact": "<completed standalone fact with all specific details>"
   }
Examples:
--- Topic 1 ---
[2024-01-07T17:24:00.000, Sun] 0.Tim: Hey John! Next month I'm off to Ireland for a
semester in Galway
[2024-01-07T17:24:01.000, Sun] 1.John: That's awesome! Where will you stay?
[2024-01-07T17:24:02.000, Sun] 2.Tim: In Galway. I also want to visit The Cliffs of Moher
[2024-01-07T17:24:03.000, Sun] 3.John: Nice! By the way, I held a benefit basketball game
last week
[visual_context: a basketball court with players and audience]
[2024-01-07T17:24:04.000, Sun] 4.Tim: Cool! I'm currently reading "The Name of the Wind" by
Patrick Rothfuss
[2024-01-07T17:24:05.000, Sun] 5.John: That sounds interesting!
--- Topic 2 ---
[2024-01-12T13:41:00.000, Fri] 6.John: Got great news! I got an endorsement with a popular
beverage company last week
[2024-01-12T13:41:01.000, Fri] 7.Tim: Congrats! That's amazing
[2024-01-12T13:41:02.000, Fri] 8.John: Thanks! By the way, Barcelona is a must-visit city
[2024-01-12T13:41:03.000, Fri] 9.Tim: I'll add it to my list!
{"data": [
  {"source_id": 0, "fact": "Tim is going to Ireland for a semester in Galway the month
after 2024-01-07."},
  {"source_id": 0, "fact": "Tim will study in Galway, Ireland the month after 2024-01-07."},
  {"source_id": 2, "fact": "Tim will stay in Galway."},
  {"source_id": 2, "fact": "Tim wants to visit The Cliffs of Moher."},
  {"source_id": 3, "fact": "John held a benefit basketball game at a basketball court with
players and audience the week before 2024-01-07."},
  {"source_id": 4, "fact": "Tim is currently reading 'The Name of the Wind' by Patrick
Rothfuss."},
  {"source_id": 4, "fact": "Tim is reading a fantasy novel."},
  {"source_id": 6, "fact": "John got an endorsement with a beverage company the week before
2024-01-12."},
  {"source_id": 8, "fact": "John recommends Barcelona as a must-visit city."},
  {"source_id": 9, "fact": "Tim has a travel list and plans to add Barcelona to it."}
]}
Reminder: Be exhaustive and ALWAYS include specific names, titles, locations, and details
in every fact.
[USER]:
--- Topic {global_topic_id} ---
{topic_text}
```

Figure 5: Factual entry extraction prompt (Part 2).

Prompts for Relational Entry Extract

```
[SYSTEM]: You are a Relational Memory Extractor.
Your task is to extract **how people relate to each other** from conversations.
Note: Another system extracts factual content (what was said).
Your focus is on the **relational and emotional dynamics** between people.
The dialogue is organized into topic segments:
--- Topic X ---
[timestamp, weekday] <source_id>.<SpeakerName>: <message>
...
Note: Messages may include visual context marked as [visual_context: ...] which provides
additional scene information.
Important Instructions:
1. **Focus on Relational Behaviors and Emotional Exchange**:
   Extract interactions showing how people relate to each other:
   - Evaluative: praise, compliment, admire, acknowledge
   - Supportive: encourage, express confidence, cheer on, offer support
   - Emotional: express gratitude, pride, happiness, excitement, congratulations
   - Engagement: ask questions, show interest, respond with curiosity
   - Agreement: agree with, align on values, share perspective
   - Responsive: share in response to another's sharing, reciprocate
2. **What to Extract vs. What to Skip**:
   Extract: "Alice praised Bob's empathy" (relational behavior)
   Extract: "Alice asked about Bob's motivation" (engagement behavior)
   Extract: "Bob expressed gratitude for Alice's support" (emotional response)
   Skip: "Bob mentioned her support group experience" (factual content only)
   Skip: "Alice said she's been painting" (factual content only)
   BUT Extract: "Alice, in turn, shared her painting as a way of connecting" (responsive behavior)
3. **Include Necessary Context**:
   When describing interactions, include enough context to make sense.
   - Extract: "Alice praised Bob's dedication to helping LGBTQ youth"
   - Not just: "Alice praised Bob"
4. **Include Temporal Information When Relevant**:
   If the relational behavior involves time-specific events or references, include that naturally.
   - "Alice empathized with Bob's job search struggles by sharing her own experience from last
year"
   - "Bob congratulated Alice on her grad school acceptance"
   For general emotional exchanges without time context, no date needed.
5. **Combine Related Interactions**:
   Merge closely related behaviors in the same message.
   - "Alice congratulated Bob on passing the interviews and expressed excitement for her future"
6. **Use "both" for Mutual Agreement**:
   When both people express similar views or bond over shared experiences.
   - "Alice and Bob both emphasized the importance of self-care"
   - Assign to source_id where the second person completes the agreement
```

Figure 6: Relational entry extraction prompt (Part 1).

Prompts for Relational Entry Extract

```
Output format:
Return JSON with key "data", containing a list of:
{
  "source_id": "<source_id>",
  "relation": "<relational description in natural language>"
}
# EXAMPLE
--- Topic 1 ---
[2024-01-15T14:20:00.000, Mon] 0.Alice: I just got accepted to grad school!
[visual_context: a woman holding an acceptance letter and smiling]
[2024-01-15T14:20:02.000, Mon] 1.Bob: Oh nice
[2024-01-15T14:20:04.000, Mon] 2.Alice: Yeah, I'm really excited about the Computer Science program
[2024-01-15T14:20:06.000, Mon] 3.Bob: That's fantastic! I'm so proud of you. What's your research
focus?
[2024-01-15T14:20:08.000, Mon] 4.Alice: Machine learning. I've been working toward this for years.
[2024-01-15T14:20:10.000, Mon] 5.Bob: You totally deserve it. I know you'll do amazing things there.
--- Topic 2 ---
[2024-01-15T14:21:00.000, Mon] 6.Alice: Thanks! That means a lot. How's your job search going?
[2024-01-15T14:21:05.000, Mon] 7.Bob: Honestly, it's been tough. Feeling pretty discouraged.
[2024-01-15T14:21:10.000, Mon] 8.Alice: I totally get that. I went through the same thing last year.
[2024-01-15T14:21:15.000, Mon] 9.Bob: Really? How did you handle it?
[2024-01-15T14:21:20.000, Mon] 10.Alice: I focused on self-care and staying connected with friends.
[2024-01-15T14:21:25.000, Mon] 11.Bob: That's helpful advice. Thanks for sharing.
[2024-01-15T14:21:30.000, Mon] 12.Alice: Of course! You're going to land something great. Let me know
if you want to talk more.
[visual_context: two people having coffee and talking]
{"data": [
  {"source_id": "3", "relation": "Bob congratulated Alice on her grad school acceptance, expressed
pride in her achievement, and showed interest by asking about her research focus."},
  {"source_id": "5", "relation": "Bob validated Alice's deservingness and expressed confidence in her
future success."},
  {"source_id": "6", "relation": "Alice expressed gratitude for Bob's support and reciprocated by
showing interest in Bob's job search."},
  {"source_id": "8", "relation": "Alice empathized with Bob's difficulties by sharing her own similar
experience from last year."},
  {"source_id": "9", "relation": "Bob showed interest in Alice's coping strategies."},
  {"source_id": "11", "relation": "Bob expressed gratitude for Alice's advice."},
  {"source_id": "12", "relation": "Alice encouraged Bob and offered ongoing support."}
]}
Reminder: Focus on relational behaviors and emotional dynamics.
[USER]:
--- Topic {global_topic_id} ---
{topic_text}
```

Figure 7: Relational entry extraction prompt (Part 2).

Prompts for Synthesize

```
[SYSTEM]: You are a professional conversation summarization assistant with temporal awareness.
[USER]: You are a professional conversation summarization assistant.
The following conversation records contain TWO types of information:
1. **Factual information**: concrete events, plans, opinions, preferences
2. **Interaction patterns**: how speakers relate to, support, and respond to each other
Both types are important and should be preserved in the summary.
Conversation Time: {bucket}
Participants: {speakers}
Conversation Records:
{aggregated_text}
Related Temporal Context (from other time periods):
{supplementary_context}
Please generate a summary with the following requirements:
CRITICAL - What to PRESERVE:
- Specific concrete details: dates, times, locations, names of things
- Key emotional transitions and psychological changes
- Concrete action plans
- Important quotes or specific expressions when they capture essential meaning
- Temporal connections: When related context reveals specific prior events or future plans
  that directly relate to current topics, integrate them naturally with timestamps
What to DO:
1. Remove redundant repetitions while keeping all key information mentioned above
2. Organize content chronologically, showing how facts and interactions unfold together
3. Highlight causal relationships (e.g., "X happened, which gave Y the courage to do Z")
4. When integrating temporal context:
   - Cite specific times if available (e.g., "on 2022 April 15...")
   - Focus on concrete connections, not general patterns
   - Weave references naturally into the narrative, don't append them as separate summary
   - Only include if it adds meaningful context to understanding current events
5. Balance factual timeline with emotional/relational dynamics
6. Use fluent, concise natural language
7. Keep length between 200-350 words
Output the summary directly without any additional explanations or format markers.
```

Figure 8: Narrative synthesis prompt for Macro Synthesis.

Prompts for Entity Extract

```
[SYSTEM]:
You are an entity extractor for conversational messages.
Input: ENTITY_TYPES, PREVIOUS_MESSAGES, CURRENT_MESSAGES (a list of entries).
Task: extract distinct entities explicitly or implicitly mentioned across the provided
CURRENT_MESSAGES.

Rules:
1) Speaker: for each entry, extract the speaker (before the colon) as an entity if present; merge
repeated mentions of the same speaker.
2) Only emit entities that appear in CURRENT_MESSAGES; use PREVIOUS_MESSAGES only for coreference
resolution.
3) Names: clear, unambiguous, lowercase_with_underscores.
4) Types: choose entity_type from ENTITY_TYPES, Default entity classification is Entity. Use this
entity type if the entity is not one of the other listed types..
5) Do not emit pronouns (you/I/he/she/they), times, dates, or pure actions/relations.

Output JSON ONLY:
{
    "entities": [
        {
            "id": <int>,                          // temporary id for this extraction round
            "name": "lower_snake_case",
            "entity_type": "string_from_ENTITY_TYPES",
            "aliases": ["..."],
            "first_entry_index": <int>         // index in CURRENT_MESSAGES where entity first
appears (0-based)
        }
    ]
}
[USER]:
"ENTITY_TYPES": {entity_types},
"PREVIOUS_MESSAGES": {previous_messages},
"CURRENT_MESSAGES": {messages},
```

Figure 9: Entity extraction prompt

Prompts for Entity Depuplicate

```
[SYSTEM]:
You decide whether a NEW ENTITY is a duplicate of any EXISTING ENTITIES. Use
PREVIOUS_MESSAGES/CURRENT_MESSAGES only for disambiguation.

Inputs include NEW_ENTITY (with id/name/entity_type/aliases/first_entry_index) and EXISTING_ENTITIES
(array with idx, name, entity_type, aliases).

Output JSON ONLY:
{
    "entity_resolutions": [
        {
            "id": <NEW_ENTITY.id>,
            "name": "best_full_name",
            "is_duplicate": true|false,
            "duplicate_idx": <int idx in EXISTING_ENTITIES or -1>,
            "duplicates": [idx1, idx2, ...]   // indices from EXISTING_ENTITIES, sorted, unique
        }
    ]
}

If unsure, set is_duplicate=false and duplicate_idx=-1 and duplicates=[].
[USER]:
"NEW_ENTITIES": [
    {
    "id": {id},
    "name": {name},
    "entity_type": {entity_type},
    "aliases": {aliases},
    "first_entry_index": {first_entry_index},
    }
],
"EXISTING_ENTITIES": {existing_entities},
"PREVIOUS_MESSAGES": {previous_messages},
"CURRENT_MESSAGES": {messages},
```

Figure 10: Entity deduplicate prompt

Prompts for Relation Extract

```
[SYSTEM]:
You extract fact triples between entities from CURRENT_MESSAGES (list of entries).

Inputs: FACT_TYPES, PREVIOUS_MESSAGES (for disambiguation only), CURRENT_MESSAGES, ENTITIES
(extracted or existing names).

Rules:
1) source and destination must be names from ENTITIES and must be distinct.
2) relationship in SCREAMING_SNAKE_CASE (e.g., GREETED, WORKS_AT). Prefer FACT_TYPES when applicable.
3) Remove duplicates/near-duplicates across the batch.

Output JSON ONLY:
{
    "facts": [
        {
            "source": "lower_snake_case",
            "relationship": "SCREAMING_SNAKE_CASE",
            "destination": "lower_snake_case",
            "fact": "concise paraphrase",
            "source_entry_index": <int>   // index of CURRENT_MESSAGES where this fact was observed
(0-based)
        }
    ]
}
[USER]:
"FACT_TYPES": {fact_types},
"PREVIOUS_MESSAGES": {previous_messages},
"CURRENT_MESSAGES": {messages},
"ENTITIES": {canonical_names},
```

Figure 11: Relation extraction prompt

Prompts for Relation Depuplicate

```
[SYSTEM]:
You deduplicate NEW FACTS (one or more) against EXISTING FACTS and detect contradictions against
FACT_INVALIDATION_CANDIDATES.

Output JSON ONLY:
{
    "result": {
        "duplicate_facts": [ { "new_fact_idx": <int>, "existing_fact_idx": <int>, "reason":
"string" }, ... ],
        "contradicted_facts": [ { "new_fact_idx": <int>, "existing_fact_idx": <int>, "reason":
"string" }, ... ],
        "fact_types": [ { "new_fact_idx": <int>, "fact_type": "TYPE_NAME" }, ... ]
    }
}

Notes:
- Indices for EXISTING_FACTS and FACT_INVALIDATION_CANDIDATES are 0-based and independent lists.
- duplicate_facts maps which new_fact (index in NEW FACTS array) duplicates which existing_fact index.
- If no duplicates/contradictions, return empty arrays.
[USER]:
"NEW_FACTS": {relations_raw},
"EXISTING_FACTS": {existing_facts},
"FACT_INVALIDATION_CANDIDATES": {fact_invalidation_candidates}
```

Figure 12: Relation deduplicate prompt

Prompts for StructMem QA

```
[SYSTEM]: You are an intelligent memory assistant tasked with retrieving accurate information
from conversation memories.
# CONTEXT:
You have access to memories from two speakers in a conversation. These memories contain
timestamped information that may be relevant to answering the question.
# INSTRUCTIONS:
1. Carefully analyze all provided memories from both speakers
2. Pay special attention to the timestamps to determine the answer
3. If the question asks about a specific event or fact, look for direct evidence in the memories
4. If the memories contain contradictory information, prioritize the most recent memory
5. If there is a question about time references (like "last year", "two months ago", etc.),
   calculate the actual date based on the memory timestamp. For example, if a memory from
   4 May 2022 mentions "went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months, or years. For example,
   convert "last year" to "2022" or "two months ago" to "March 2023" based on the memory
   timestamp. Ignore the reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do not confuse character
   names mentioned in memories with the actual users who created those memories.
8. The answer should be less than 5-6 words.
# APPROACH (Think step by step):
1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the question
4. If the answer requires calculation (e.g., converting relative time references), show your work
5. Formulate a precise, concise answer based solely on the evidence in the memories
6. Double-check that your answer directly addresses the question asked
7. Ensure your final answer is specific and avoids vague time references
Memories for user {speaker_1_name}:
{speaker_1_memories}
Memories for user {speaker_2_name}:
{speaker_2_memories}
Session summaries:
{session_summaries}
Question: {question}
Answer:
```

Figure 13: Question answering prompt for StructMem system.

Prompts for Flat Memory system QA

```
[SYSTEM]: You are an intelligent memory assistant tasked with retrieving accurate information
from conversation memories.
# CONTEXT:
You have access to memories from two speakers in a conversation. These memories contain
timestamped information that may be relevant to answering the question.
# INSTRUCTIONS:
1. Carefully analyze all provided memories from both speakers
2. Pay special attention to the timestamps to determine the answer
3. If the question asks about a specific event or fact, look for direct evidence in the memories
4. If the memories contain contradictory information, prioritize the most recent memory
5. If there is a question about time references (like "last year", "two months ago", etc.),
   calculate the actual date based on the memory timestamp. For example, if a memory from
   4 May 2022 mentions "went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months, or years. For example,
   convert "last year" to "2022" or "two months ago" to "March 2023" based on the memory
   timestamp. Ignore the reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do not confuse character
   names mentioned in memories with the actual users who created those memories.
8. The answer should be less than 5-6 words.
# APPROACH (Think step by step):
1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the question
4. If the answer requires calculation (e.g., converting relative time references), show your work
5. Formulate a precise, concise answer based solely on the evidence in the memories
6. Double-check that your answer directly addresses the question asked
7. Ensure your final answer is specific and avoids vague time references
Memories for user {speaker_1_name}:
{speaker_1_memories}
Memories for user {speaker_2_name}:
{speaker_2_memories}
Question: {question}
Answer:
```

Figure 14: Question answering prompt for flat memory systems.

Prompts for Graph Memory system QA

```
[SYSTEM]: You are an intelligent memory assistant tasked with retrieving accurate information
from conversation memories.
# CONTEXT:
You have access to memories from two speakers in a conversation. These memories contain
timestamped information that may be relevant to answering the question. You also have access to
knowledge graph relations for each user, showing connections between entities, concepts, and
events relevant to that user.
# INSTRUCTIONS:
1. Carefully analyze all provided memories from both speakers
2. Pay special attention to the timestamps to determine the answer
3. If the question asks about a specific event or fact, look for direct evidence in the memories
4. If the memories contain contradictory information, prioritize the most recent memory
5. If there is a question about time references (like "last year", "two months ago", etc.),
calculate the actual date based on the memory timestamp. For example, if a memory from 4 May 2022
mentions "went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months, or years. For example,
convert "last year" to "2022" or "two months ago" to "March 2023" based on the memory timestamp.
Ignore the reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do not confuse character names
mentioned in memories with the actual users who created those memories.
8. The answer should be less than 5-6 words.
9. Use the knowledge graph relations to understand the user's knowledge network and identify
important relationships between entities in the user's world.
# APPROACH (Think step by step):
1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the question
4. If the answer requires calculation (e.g., converting relative time references), show your work
5. Analyze the knowledge graph relations to understand the user's knowledge context
6. Formulate a precise, concise answer based solely on the evidence in the memories
7. Double-check that your answer directly addresses the question asked
8. Ensure your final answer is specific and avoids vague time references
Memories for user {{speaker_1_user_id}}:
{{speaker_1_memories}}
Relations for user {{speaker_1_user_id}}:
{{speaker_1_graph_memories}}
Memories for user {{speaker_2_user_id}}:
{{speaker_2_memories}}
Relations for user {{speaker_2_user_id}}:
{{speaker_2_graph_memories}}
Question: {{question}}
Answer:
```

Figure 15: Question answering prompt for graph-based memory systems.

Prompts for Evaluating Answer

```
[USER]: Your task is to label an answer to a question as 'CORRECT' or 'WRONG'. You will be
given the following data:
    (1) a question (posed by one user to another user),
    (2) a 'gold' (ground truth) answer,
    (3) a generated answer
which you will score as CORRECT/WRONG.

The point of the question is to ask about something one user should know about the other user
based on their prior conversations.
The gold answer will usually be a concise and short answer that includes the referenced topic,
for example:
Question: Do you remember what I got the last time I went to Hawaii?
Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous with your grading - as
long as it touches on the same topic as the gold answer, it should be counted as CORRECT.

For time related questions, the gold answer will be a specific date, month, year, etc. The
generated answer might be much longer or use relative time references (like "last Tuesday" or
"next month"), but you should be generous with your grading - as long as it refers to the
same date or time period as the gold answer, it should be counted as CORRECT. Even if the
format differs (e.g., "May 7th" vs "7 May"), consider it CORRECT if it's the same date.

Now it's time for the real question:
Question: {question}
Gold answer: {gold_answer}
Generated answer: {generated_answer}

First, provide a short (one sentence) explanation of your reasoning, then finish with CORRECT
or WRONG.
Do NOT include both CORRECT and WRONG in your response, or it will break the evaluation
script.

Just return the label CORRECT or WRONG in a json format with the key as "label".
```

Figure 16: Evaluate prompt for assessing response quality.